a9 FIG. 10 illustrates the Selection Properties dialog screens 116 used in accordance with the software program of the present invention.

REMARKS

The specification has been amended to correct certain informalities noted by the Applicant in reviewing the application. In particular, reference numbers from the drawing figures have been added to the appropriate sections of the specification to more clearly and particularly define the subject matter of the invention. No new matter has been added. The specification is believed to be in acceptable form.

The Applicant believes that the application is now in condition for allowance, and respectfully requests consideration of the application as amended. The Applicant respectfully requests that the Examiner telephone the undersigned in the event a telephone conference would expedite prosecution of the application.

Respectfully submitted,

GODFREY & KAHN, S.C.

Dated: July 20, 2004

By: William K. Baxter

William K. Baxter
Reg. No. 41,606

Attorneys of Record for Applicant
GODFREY & KAHN, S.C.
780 North Water Street
Milwaukee, WI 53202-3590
(414) 273-3500

MW535593_1.DOC

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Specification:

In accordance with 37 C.F.R. § 1.121(b)(1)(ii), the following is a marked-up version of the replacement paragraphs of the specification.

The paragraph beginning at page 6, line 6, has been amended as follows:

The lowest level component is the compression/extraction engine 40. The compression/extraction component 40 consists of the actual compression, extraction, and ~~crc-32~~ CRC-32 algorithms in converter, reader, writer components 42, 44, 46, 48 interfacing with data object and archive files 34, 36 in the file management component 30. ~~These~~ The compression, extraction and CRC-32 algorithms are written as a set of portable C language routines, with higher level C++ routines interfacing with the higher level file management component 30. The file management component 30 consists of the central directory 32 which holds a cached tree-like structure of the archive independent of the actual archive type. Actual archive implementation is used by the central directory 22 to read/write data to the archives 34 and the user interface 20. The central directory 32 consists of folder objects and file objects 36. A services object 38 is also part of the file management component 30. The services object 38 acts as a helper interface between the user interface component 20 and the file management component 30. The user interface 20 consists of a shell 22, a graphical user interface (GUI) 24, and a call level interface (CLI) 26.

The paragraph beginning at page 6, line 27, has been amended as follows:

FIGS. 2a–2e illustrate the different compression and extraction used by the present invention. In FIG. 2a, regular compression and extraction chains 50 are shown. These chains 50 include chains for compression 52, extraction 54, compression with encryption 56, and extraction with decryption 58. In FIG. 2b, compression data chains 60 are shown, including the use of a generic converter involving no compression for as is files 62 and encrypted files 64. These compression data chains 60 further include chains for compressed files 66 and compressed encrypted files 68. In FIG. 2c, data here compression chains 70 are shown. These compress data here chains 70 include chains for as is files 72, encrypted files 74, compressed files 76 and compressed encrypted files 78. In FIG. 2d, GetData extraction chains 80 for as is files 82, not encrypted files 84, uncompressed files 86 and decrypted uncompressed files 88 are shown. In FIG. 2e, GetDataHere extraction chains 90 for as is files 92, not encrypted files 94, uncompressed files 96 and decrypted uncompressed files 98 are shown.

The paragraph beginning at page 7, line 13, has been amended as follows:

FIG. 3 displays a right-click context menu ~~50~~ 100 of the present invention which may be used to open, modify and extract files from an existing zip file, or create a new zip file. In opening a zip file, a user may simply double-click the file to view the contents of the file. Alternatively, the following is an example of the steps one might follow to open and view the contents of a zip file. First, the zip archive to be opened is located by using Windows Explorer. Then, the user right-clicks on the zip file ~~he~~ he/she wants to open. A context menu appears. PKZIP|Explore is selected. The contents of the zip file will be displayed in the right pane under the archive manager. As another alternative, a user may select PKZIP|Explore PKZIP Folder to

create a folder shortcut under the current folder, and display the contents of the zip file via this folder.

The paragraph beginning at page 7, line 22, has been amended as follows:

To extract individual files and/or folders archived in a zip file, a user opens the zip file in Explorer as discussed above and invokes the ~~extract~~ Extract dialog 110, by selecting the PKZIP|Extract menu item in the right-click context menu. The Extract dialog 110 appears, FIG. 7, allowing the user to manually specify a destination directory. Alternatively, a user may select PKZIP|Extract Here to extract the contents of the archive into the directory where the zip archive resides. To create a directory (e.g., "Test") under the directory where the zip archive resides, and extract all files in that directory, the user selects the ~~"Extract to"~~ PKZIP|Extract-To menu item. Alternatively, files may be extracted using a drag and drop operation. The user highlights the files and/or folders he/she wishes to extract, drags the files to a destination, and drops the files in the enabled destination. The files and/or folders will be automatically extracted into the drop destination. As the extraction process proceeds, the progress is displayed in a progress dialog 102, as shown in FIG. 4. If there is an error encountered during the extraction process, the error is indicated in the progress dialog 102 and a ~~log~~ Log dialog 104, shown in FIG. 9.

The paragraph beginning at page 8, line 7, has been amended as follows:

The present invention also allows a user to create a new zip file. The following is an example of the steps one might follow to create a new zip file. First, the user highlights the files and/or folders he wishes to archive. The user then clicks his right mouse button to bring up the context menu. PKZIP|Compress is then selected. The ~~"Save as"~~ Save As dialog 104 appears,

FIG. 5. A name and destination are specified for the zip file, and the ~~save~~ Save button is clicked to proceed. The progress dialog 102 appears to monitor completion and to indicate errors in the process. The new zip file should now reside in the specified destination directory. A user may alternatively create a new zip file by other means as well. A user may create a new folder in an archive by selecting the New Folder menu toolbar item and specifying a folder name as desired.

The paragraph beginning at page 8, line 24, has been amended as follows:

The following is an example of the steps one might follow to modify an existing zip file. A user first locates and opens the zip file he/she wishes to modify. Next, the files and/or folders to add to the archive are specified by clicking the ~~add~~ Add toolbar button, thereby invoking the ~~add~~ Add dialog 112, illustrated in FIG. 8, or by dragging the files and/or folders from their source and dropping them at a destination. A user may alternately use the copy and paste operation to specify files and/or folders to add to the archive. The program will display an add icon (such as plus symbol) indicating that these files and/or folders are to be added when the archive is saved.

The paragraph beginning at page 9, line 11, has been amended as follows:

FIGS. 6a and 6b display modifications and additions to the Explorer toolbar buttons 106 and menu items 108 used in the present invention.

The paragraph beginning at page 10, line 15, has been amended as follows:

The present invention also allows the user to digitally sign and encrypt the individual files archived in a zip file as well as the central end directory, and subsequently to authenticate

and decrypt those files upon extraction. The signing and encrypting functionality is based on PKCS No. 7, and related public key encryption standards and is therefore compatible with security functionality in other applications such as Microsoft's Internet Explorer. Signing a zip file allows one to detect whether a zip file's integrity has been compromised. Encrypting a file denies access to the file's contents by unauthorized users. These features are illustrated in the ZIP Components Properties and Digital ID Properties dialog screens 118 as shown in FIG. 11.

The following new paragraph has been added between the paragraph beginning at page 10, line 22 and the paragraph beginning at page 10, line 25:

--FIG. 10 illustrates the Selection Properties dialog screens 116 used in accordance with the software program of the present invention.--

MW535593_1.DOC